**Hadoop Balancer Utility-**

HDFS data might not always be placed uniformly across DataNodes. One common reason is addition of new DataNodes to an existing cluster. HDFS provides a balancer utility that analyzes block placement and balances data across the DataNodes. It moves blocks until the cluster is deemed to be balanced, which means that the utilization of every DataNode (ratio of used space on the node to total capacity of the node) differs from the utilization of the cluster (ratio of used space on the cluster to total capacity of the cluster) by no more than a given threshold percentage. The balancer does not balance between individual volumes on a single DataNode.

This is the balancer of hadoop and it is generally managed by the Hadoop Administrator.  
To run a cluster balancing utility we run the following command:  
$ hadoop balancer [-threshold ]  
where -threshold is percentage of disk capacity. This overwrites the default threshold.

**Distcp To Copy Data –**

DistCp (distributed copy) is a tool used for large inter/intra-cluster copying. It uses MapReduce to effect its distribution, error handling and recovery, and reporting. It expands a list of files and directories into input to map tasks, each of which will copy a partition of the files specified in the source list.

DistCp (distributed copy) is a tool used for large inter/intra-cluster copying. It uses MapReduce to effect its distribution, error handling and recovery, and reporting. It expands a list of files and directories into input to map tasks, each of which will copy a partition of the files specified in the source list. Its MapReduce pedigree has endowed it with some quirks in both its semantics and execution. The purpose of this document is to offer guidance for common tasks and to elucidate its model.

The distributed copy command, [distcp](http://hadoop.apache.org/docs/current/hadoop-distcp/DistCp.html), is a general utility for copying large data sets between distributed filesystems within and across clusters. The distcp command submits a regular MapReduce job that performs a file-by-file copy.

To see the distcp command options, run the built-in help:

$ hadoop distcp hdfs://nn1:8020/source hdfs://nn2:8020/destination

**Setup Trash Directory-**

The Hadoop trash feature helps prevent accidental deletion of files and directories. When you delete a file in HDFS, the file is not immediately expelled from HDFS. Deleted files are first moved to the /user/<username>/.Trash/Current directory, with their original filesystem path being preserved. After a user-configurable period of time (fs.trash.interval), a process known as trash checkpointing renames the Current directory to the current timestamp, that is, /user/<username>/.Trash/<timestamp>. The checkpointing process also checks the rest of the .Trash directory for any existing timestamp directories and removes them from HDFS permanently. You can restore files and directories in the trash simply by moving them to a location outside the .Trash directory.

HDFS trash is just like the Recycle Bin in Windows operating systems. Its purpose is to prevent you from unintentionally deleting something. You can enable this feature by setting this property:

fs.trash.interval

with a number greater than 0 in core-site.xml. After the trash feature is enabled, when you remove something from HDFS by using the **rm** command, files or directories will not be wiped out immediately; instead, they will be moved to a trash directory (/user/${username}/.Trash, for example). In the preceding output:

* Deletion interval specifies how long (in minutes) a checkpoint will be expired before it is deleted. It is the value of fs.trash.interval. The NameNode runs a thread to periodically remove expired checkpoints from the file system.
* Emptier interval specifies how long (in minutes) the NameNode waits before running a thread to manage checkpoints. The NameNode deletes checkpoints that are older than fs.trash.interval and creates a new checkpoint from /user/${username}/.Trash/Current. This frequency is determined by the value of fs.trash.checkpoint.interval, and it must *not* be greater than the deletion interval. This ensures that in an emptier window, there are one or more checkpoints in the trash.

For example, set

fs.trash.interval = 360 (deletion interval = 6 hours)  
fs.trash.checkpoint.interval = 60 (emptier interval = 1 hour)

This causes the NameNode to create a new checkpoint every hour and to delete checkpoints that have existed longer than 6 hours.

A checkpoint is merely a directory under the user trash that is used to store all files or directories that were deleted before the checkpoint is created. If you want to take a look at the trash directory, you can see it at /user/${username}/.Trash/{timestamp\_of\_checkpoint\_creation}.

The first thing that comes to your mind is “Just delete the entire trash directory; that would remove everything”. True, that is always an option. But you have a better option. HDFS provides a command line utility to do that:

hadoop fs -expunge

This command causes the NameNode to permanently delete files from the trash that are older than the threshold, instead of waiting for the next emptier window. It immediately removes expired checkpoints from the file system.

For a production environment, it is recommended that you enable trash to avoid unexpected removal operations. Enabling trash provides a chance to recover data from operational or user errors. But it is also important to set appropriate values for **fs.trash.interval** and **fs.trash.checkpoint.interval** to make trash work the way you expect it to work. For example, if you need to frequently upload and delete files from the HDFS, you probably want to set **fs.trash.interval** to a smaller value, otherwise the checkpoints would take up too much space.

Keep in mind that when trash is enabled and you remove some files, HDFS capacity does not increase because files are not truly deleted. The HDFS does not reclaim the space unless the files are removed from the trash, which occurs only after checkpoints are expired. Sometimes you might want to temporarily disable trash when deleting files; in this case, you can run the **rm** command with the **-skipTrash** option. For example:

hadoop fs -rm -skipTrash /path/to/permanently/delete

**Checking The Safemode Of Name Node-**

|  |  |
| --- | --- |
|  | **Safemode** is an **HDFS** state in which the file system is mounted read-only; no replication is performed, nor can files be created or deleted. This is automatically entered as the **NameNode** starts, to allow all **DataNodes** time to check in with the NameNode and announce which blocks they hold, before the **NameNode** determines which blocks are under-replicated, etc. The **NameNode** waits until a specific percentage of the blocks are present and accounted-for; this is controlled in the configuration by the **dfs.safemode.threshold.pct parameter**. After this **threshold** is met, **safemode** is automatically exited, and HDFS allows normal operations.  Below command forces the NameNode to exit safemode  hdfs dfsadmin -safemode leave  Based on the report, It seems that Resource are low on NN. Add or free up more resources then turn off **safe mode** manually. **If you turn off safe mode before adding more resources or freeing up resource, the NameNode will immediately return to safe mode**. |